

1 --24. A process for assigning tasks in a multiprocessor digital data processing system
2 having a preemptive operating system, and a given number of processors capable of processing
3 said tasks in parallel, comprising dividing said processors (20a-21a, 20b-22b, 20c) in at least one
4 preliminary phase into groups (Ga, Gb, Gc) each group comprising predetermined numbers of
5 processors, and each of said processor groups being associated with an elementary queue (5a, 5b,
6 5c), each of the tasks being associated with one of the processors associated with said elementary
7 queue (5a, 5b, 5c), and storing a predetermined number of tasks to be processed in a given order
8 of priority.

A2
S1
B1
25. A process according to claim 24, characterized in that said groups each comprise
an identical number of processors (200-203, 210-213).

26. A process according to claim 24, comprising generating a series of tests and
measurements in an additional preliminary phase for determining the number of processors in
each group and the number of groups for achieving the best performance of said system.

27. A process according to claim 24, wherein the architecture of said system is of the
non-uniform memory access type (NUMA), and the system (1) is constituted by a predetermined
number of modules (M0, M1) linked to one another, each comprising a given number of
processors (200-203, 210-213) and storage means, each of said modules (M0, M1) constituting
one of said groups, each module being associated with one of said elementary queues of an
associated processor.

1 28. A process according to claim 24, further comprising associating each of said
2 processors with a first data structure for identification of the associated processor, said first data
3 structure comprises at least one first set of pointers (p200 through p203), associating said first set
4 of pointers with one of said elementary queues (5a, 5b), associating each of said elementary
5 queues (5a, 5b) with a second data structure, said second data structure having at least one
6 second set of pointers (pp5a, pp5b), associating said second data structure with one of said

7 processor groups (200-201, 202-203), storing all of the tasks to be processed (T_1 through T_{10}) in
8 said system (1) in a table (4), each of said second data structures of the elementary queues (5a,
9 5b) further comprising a third set of pointers (pT_1, pT_5, pT_{10}), said third set of pointers each
10 associating elementary queues (5a, 5b) with one of said tasks (T_1 through T_{10}) stored in the
11 table (4) or with a series of concatenated tasks, and associating each of said tasks (T_1 through
12 T_{10}) of the table (4) with a third data structure that comprises a fourth set of pointers ($p5a_1$
13 through $p5a_4, p5b_1$ through $p5b_{10}$) said fourth set of pointers associating third data structure
14 with one of said elementary queues (5a, 5b).

A
1 29. A process according to claim 24, further comprising distributing said tasks among
2 said elementary queues (5a, 5b) in at least one additional phase by searching, when a new task to
3 be processed (T_z) is created, for a queue with the lightest load (5y) among all of said elementary
4 queues (5a, 5x, 5y, 5p) of said system (1) and assigning said new task to said elementary queue
5 with the lightest load so as to balance the global load of said system (1) among said elementary
6 queues (5a, 5x, 5y, 5p).

1 30. A process according to claim 29, further comprising performing said distribution
2 of tasks by determining a composite load parameter associated with each of said elementary
3 queues (5a, 5x, 5y, 5p) associating each processor (2a, 2x, 2y, 2p) with a memory (Mem_a,
4 Mem_x, Mem_y, Mem_p), calculating said composite load parameter as the sum of the load of a
5 processor or a processor group associated with said elementary queue and the load of the
6 memory associated with said processor or processor group.

1 31. A process according to claim 29, further comprising checking in a preliminary
2 step whether said task (T_z) is linked to one of said elementary queues (5a, 5x, 5y, 5p), and when
3 said test is positive, assigning said linked task to the elementary queue.

1 32. A process according to claim 24, further comprising at least one additional phase
2 and searching for a remote elementary queue (5y) that is not empty when one of said elementary

3 queues ($5q$) associated with one of said processor groups ($2q$) is empty of executable tasks
4 selecting in said empty elementary queue ($5y$) a task executable by one of said processors ($2q$) of
5 said processor group associated with the empty elementary queue ($5q$) and transmitting said
6 selected task to said one of said processor ($2q$) for processing so as to globally balance the
7 processing of said tasks in said system (1).

1 33. A process according to claim 32, characterized in that said non-empty elementary
2 queue ($5y$) has a predetermined minimal occupation threshold.

A₂
Sub
B₁
B₂
B₃
B₄
B₅
B₆
B₇
B₈
B₉
B₁₀
B₁₁
B₁₂
B₁₃
B₁₄
B₁₅
B₁₆
B₁₇
B₁₈
B₁₉
B₂₀
B₂₁
B₂₂
B₂₃
B₂₄
B₂₅
B₂₆
B₂₇
B₂₈
B₂₉
B₃₀
B₃₁
B₃₂
B₃₃
B₃₄
B₃₅
B₃₆
B₃₇
B₃₈
B₃₉
B₄₀
B₄₁
B₄₂
B₄₃
B₄₄
B₄₅
B₄₆
B₄₇
B₄₈
B₄₉
B₅₀
B₅₁
B₅₂
B₅₃
B₅₄
B₅₅
B₅₆
B₅₇
B₅₈
B₅₉
B₆₀
B₆₁
B₆₂
B₆₃
B₆₄
B₆₅
B₆₆
B₆₇
B₆₈
B₆₉
B₇₀
B₇₁
B₇₂
B₇₃
B₇₄
B₇₅
B₇₆
B₇₇
B₇₈
B₇₉
B₈₀
B₈₁
B₈₂
B₈₃
B₈₄
B₈₅
B₈₆
B₈₇
B₈₈
B₈₉
B₉₀
B₉₁
B₉₂
B₉₃
B₉₄
B₉₅
B₉₆
B₉₇
B₉₈
B₉₉
B₁₀₀
B₁₀₁
B₁₀₂
B₁₀₃
B₁₀₄
B₁₀₅
B₁₀₆
B₁₀₇
B₁₀₈
B₁₀₉
B₁₁₀
B₁₁₁
B₁₁₂
B₁₁₃
B₁₁₄
B₁₁₅
B₁₁₆
B₁₁₇
B₁₁₈
B₁₁₉
B₁₂₀
B₁₂₁
B₁₂₂
B₁₂₃
B₁₂₄
B₁₂₅
B₁₂₆
B₁₂₇
B₁₂₈
B₁₂₉
B₁₃₀
B₁₃₁
B₁₃₂
B₁₃₃
B₁₃₄
B₁₃₅
B₁₃₆
B₁₃₇
B₁₃₈
B₁₃₉
B₁₄₀
B₁₄₁
B₁₄₂
B₁₄₃
B₁₄₄
B₁₄₅
B₁₄₆
B₁₄₇
B₁₄₈
B₁₄₉
B₁₅₀
B₁₅₁
B₁₅₂
B₁₅₃
B₁₅₄
B₁₅₅
B₁₅₆
B₁₅₇
B₁₅₈
B₁₅₉
B₁₆₀
B₁₆₁
B₁₆₂
B₁₆₃
B₁₆₄
B₁₆₅
B₁₆₆
B₁₆₇
B₁₆₈
B₁₆₉
B₁₇₀
B₁₇₁
B₁₇₂
B₁₇₃
B₁₇₄
B₁₇₅
B₁₇₆
B₁₇₇
B₁₇₈
B₁₇₉
B₁₈₀
B₁₈₁
B₁₈₂
B₁₈₃
B₁₈₄
B₁₈₅
B₁₈₆
B₁₈₇
B₁₈₈
B₁₈₉
B₁₉₀
B₁₉₁
B₁₉₂
B₁₉₃
B₁₉₄
B₁₉₅
B₁₉₆
B₁₉₇
B₁₉₈
B₁₉₉
B₂₀₀
B₂₀₁
B₂₀₂
B₂₀₃
B₂₀₄
B₂₀₅
B₂₀₆
B₂₀₇
B₂₀₈
B₂₀₉
B₂₁₀
B₂₁₁
B₂₁₂
B₂₁₃
B₂₁₄
B₂₁₅
B₂₁₆
B₂₁₇
B₂₁₈
B₂₁₉
B₂₂₀
B₂₂₁
B₂₂₂
B₂₂₃
B₂₂₄
B₂₂₅
B₂₂₆
B₂₂₇
B₂₂₈
B₂₂₉
B₂₃₀
B₂₃₁
B₂₃₂
B₂₃₃
B₂₃₄
B₂₃₅
B₂₃₆
B₂₃₇
B₂₃₈
B₂₃₉
B₂₄₀
B₂₄₁
B₂₄₂
B₂₄₃
B₂₄₄
B₂₄₅
B₂₄₆
B₂₄₇
B₂₄₈
B₂₄₉
B₂₅₀
B₂₅₁
B₂₅₂
B₂₅₃
B₂₅₄
B₂₅₅
B₂₅₆
B₂₅₇
B₂₅₈
B₂₅₉
B₂₆₀
B₂₆₁
B₂₆₂
B₂₆₃
B₂₆₄
B₂₆₅
B₂₆₆
B₂₆₇
B₂₆₈
B₂₆₉
B₂₇₀
B₂₇₁
B₂₇₂
B₂₇₃
B₂₇₄
B₂₇₅
B₂₇₆
B₂₇₇
B₂₇₈
B₂₇₉
B₂₈₀
B₂₈₁
B₂₈₂
B₂₈₃
B₂₈₄
B₂₈₅
B₂₈₆
B₂₈₇
B₂₈₈
B₂₈₉
B₂₉₀
B₂₉₁
B₂₉₂
B₂₉₃
B₂₉₄
B₂₉₅
B₂₉₆
B₂₉₇
B₂₉₈
B₂₉₉
B₃₀₀
B₃₀₁
B₃₀₂
B₃₀₃
B₃₀₄
B₃₀₅
B₃₀₆
B₃₀₇
B₃₀₈
B₃₀₉
B₃₁₀
B₃₁₁
B₃₁₂
B₃₁₃
B₃₁₄
B₃₁₅
B₃₁₆
B₃₁₇
B₃₁₈
B₃₁₉
B₃₂₀
B₃₂₁
B₃₂₂
B₃₂₃
B₃₂₄
B₃₂₅
B₃₂₆
B₃₂₇
B₃₂₈
B₃₂₉
B₃₃₀
B₃₃₁
B₃₃₂
B₃₃₃
B₃₃₄
B₃₃₅
B₃₃₆
B₃₃₇
B₃₃₈
B₃₃₉
B₃₄₀
B₃₄₁
B₃₄₂
B₃₄₃
B₃₄₄
B₃₄₅
B₃₄₆
B₃₄₇
B₃₄₈
B₃₄₉
B₃₅₀
B₃₅₁
B₃₅₂
B₃₅₃
B₃₅₄
B₃₅₅
B₃₅₆
B₃₅₇
B₃₅₈
B₃₅₉
B₃₆₀
B₃₆₁
B₃₆₂
B₃₆₃
B₃₆₄
B₃₆₅
B₃₆₆
B₃₆₇
B₃₆₈
B₃₆₉
B₃₇₀
B₃₇₁
B₃₇₂
B₃₇₃
B₃₇₄
B₃₇₅
B₃₇₆
B₃₇₇
B₃₇₈
B₃₇₉
B₃₈₀
B₃₈₁
B₃₈₂
B₃₈₃
B₃₈₄
B₃₈₅
B₃₈₆
B₃₈₇
B₃₈₈
B₃₈₉
B₃₉₀
B₃₉₁
B₃₉₂
B₃₉₃
B₃₉₄
B₃₉₅
B₃₉₆
B₃₉₇
B₃₉₈
B₃₉₉
B₄₀₀
B₄₀₁
B₄₀₂
B₄₀₃
B₄₀₄
B₄₀₅
B₄₀₆
B₄₀₇
B₄₀₈
B₄₀₉
B₄₁₀
B₄₁₁
B₄₁₂
B₄₁₃
B₄₁₄
B₄₁₅
B₄₁₆
B₄₁₇
B₄₁₈
B₄₁₉
B₄₂₀
B₄₂₁
B₄₂₂
B₄₂₃
B₄₂₄
B₄₂₅
B₄₂₆
B₄₂₇
B₄₂₈
B₄₂₉
B₄₃₀
B₄₃₁
B₄₃₂
B₄₃₃
B₄₃₄
B₄₃₅
B₄₃₆
B₄₃₇
B₄₃₈
B₄₃₉
B₄₄₀
B₄₄₁
B₄₄₂
B₄₄₃
B₄₄₄
B₄₄₅
B₄₄₆
B₄₄₇
B₄₄₈
B₄₄₉
B₄₅₀
B₄₅₁
B₄₅₂
B₄₅₃
B₄₅₄
B₄₅₅
B₄₅₆
B₄₅₇
B₄₅₈
B₄₅₉
B₄₆₀
B₄₆₁
B₄₆₂
B₄₆₃
B₄₆₄
B₄₆₅
B₄₆₆
B₄₆₇
B₄₆₈
B₄₆₉
B₄₇₀
B₄₇₁
B₄₇₂
B₄₇₃
B₄₇₄
B₄₇₅
B₄₇₆
B₄₇₇
B₄₇₈
B₄₇₉
B₄₈₀
B₄₈₁
B₄₈₂
B₄₈₃
B₄₈₄
B₄₈₅
B₄₈₆
B₄₈₇
B₄₈₈
B₄₈₉
B₄₉₀
B₄₉₁
B₄₉₂
B₄₉₃
B₄₉₄
B₄₉₅
B₄₉₆
B₄₉₇
B₄₉₈
B₄₉₉
B₅₀₀
B₅₀₁
B₅₀₂
B₅₀₃
B₅₀₄
B₅₀₅
B₅₀₆
B₅₀₇
B₅₀₈
B₅₀₉
B₅₁₀
B₅₁₁
B₅₁₂
B₅₁₃
B₅₁₄
B₅₁₅
B₅₁₆
B₅₁₇
B₅₁₈
B₅₁₉
B₅₂₀
B₅₂₁
B₅₂₂
B₅₂₃
B₅₂₄
B₅₂₅
B₅₂₆
B₅₂₇
B₅₂₈
B₅₂₉
B₅₃₀
B₅₃₁
B₅₃₂
B₅₃₃
B₅₃₄
B₅₃₅
B₅₃₆
B₅₃₇
B₅₃₈
B₅₃₉
B₅₄₀
B₅₄₁
B₅₄₂
B₅₄₃
B₅₄₄
B₅₄₅
B₅₄₆
B₅₄₇
B₅₄₈
B₅₄₉
B₅₅₀
B₅₅₁
B₅₅₂
B₅₅₃
B₅₅₄
B₅₅₅
B₅₅₆
B₅₅₇
B₅₅₈
B₅₅₉
B₅₆₀
B₅₆₁
B₅₆₂
B₅₆₃
B₅₆₄
B₅₆₅
B₅₆₆
B₅₆₇
B₅₆₈
B₅₆₉
B₅₇₀
B₅₇₁
B₅₇₂
B₅₇₃
B₅₇₄
B₅₇₅
B₅₇₆
B₅₇₇
B₅₇₈
B₅₇₉
B₅₈₀
B₅₈₁
B₅₈₂
B₅₈₃
B₅₈₄
B₅₈₅
B₅₈₆
B₅₈₇
B₅₈₈
B₅₈₉
B₅₉₀
B₅₉₁
B₅₉₂
B₅₉₃
B₅₉₄
B₅₉₅
B₅₉₆
B₅₉₇
B₅₉₈
B₅₉₉
B₆₀₀
B₆₀₁
B₆₀₂
B₆₀₃
B₆₀₄
B₆₀₅
B₆₀₆
B₆₀₇
B₆₀₈
B₆₀₉
B₆₁₀
B₆₁₁
B₆₁₂
B₆₁₃
B₆₁₄
B₆₁₅
B₆₁₆
B₆₁₇
B₆₁₈
B₆₁₉
B₆₂₀
B₆₂₁
B₆₂₂
B₆₂₃
B₆₂₄
B₆₂₅
B₆₂₆
B₆₂₇
B₆₂₈
B₆₂₉
B₆₃₀
B₆₃₁
B₆₃₂
B₆₃₃
B₆₃₄
B₆₃₅
B₆₃₆
B₆₃₇
B₆₃₈
B₆₃₉
B₆₄₀
B₆₄₁
B₆₄₂
B₆₄₃
B₆₄₄
B₆₄₅
B₆₄₆
B₆₄₇
B₆₄₈
B₆₄₉
B₆₅₀
B₆₅₁
B₆₅₂
B₆₅₃
B₆₅₄
B₆₅₅
B₆₅₆
B₆₅₇
B₆₅₈
B₆₅₉
B₆₆₀
B₆₆₁
B₆₆₂
B₆₆₃
B₆₆₄
B₆₆₅
B₆₆₆
B₆₆₇
B₆₆₈
B₆₆₉
B₆₇₀
B₆₇₁
B₆₇₂
B₆₇₃
B₆₇₄
B₆₇₅
B₆₇₆
B₆₇₇
B₆₇₈
B₆₇₉
B₆₈₀
B₆₈₁
B₆₈₂
B₆₈₃
B₆₈₄
B₆₈₅
B₆₈₆
B₆₈₇
B₆₈₈
B₆₈₉
B₆₉₀
B₆₉₁
B₆₉₂
B₆₉₃
B₆₉₄
B₆₉₅
B₆₉₆
B₆₉₇
B₆₉₈
B₆₉₉
B₇₀₀
B₇₀₁
B₇₀₂
B₇₀₃
B₇₀₄
B₇₀₅
B₇₀₆
B₇₀₇
B₇₀₈
B₇₀₉
B₇₁₀
B₇₁₁
B₇₁₂
B₇₁₃
B₇₁₄
B₇₁₅
B₇₁₆
B₇₁₇
B₇₁₈
B₇₁₉
B₇₂₀
B₇₂₁
B₇₂₂
B₇₂₃
B₇₂₄
B₇₂₅
B₇₂₆
B₇₂₇
B₇₂₈
B₇₂₉
B₇₃₀
B₇₃₁
B₇₃₂
B₇₃₃
B₇₃₄
B₇₃₅
B₇₃₆
B₇₃₇
B₇₃₈
B₇₃₉
B₇₄₀
B₇₄₁
B₇₄₂
B₇₄₃<

3 additional phase and when an unbalanced state of said system (1) is determined, selectively
4 moving tasks from at least one elementary queue with a heavier load (5x) to an elementary queue
5 with a lighter load (5y).

1 38. A process according to claim 37 comprising discontinuing the step of selectively
2 moving tasks when said imbalance is below a certain threshold.

1 39. A process according to claim 37 wherein all or some of said tasks belong to
2 multitask processes, and each multitask process requires a given memory size and workload,
3 further comprising measuring workloads and memory sizes, in the system and selecting the
A24 process requiring the greatest workload and the smallest memory size, and moving all the tasks
of said selected process to the elementary queue with the lightest load (5y).

Su
B1
2
3
4
5
6
7 40. A process according to claim 39, characterized in that it comprises a preliminary
step of checking whether all tasks of said multitask process that must be moved belong to the
elementary queue set with the heaviest load (5x) and whether any task is linked to any of said
groups.

1 41.. A process according to claim 24 characterized in that said preemptive operating
2 system is of the "UNIX" type.

1 42. Architecture for a multiprocessor digital data processing system comprising a
2 given number of processors for implementing a process for assigning tasks to be processed to
3 said processors, said system having a preemptive operating system and a given number of
4 processors capable of processing said task in parallel, said processors (20a-21a, 20b-22b, 20c)
5 being divided into groups (Ga, Gb, Gc), and an elementary queue (5a, 5b, 5c) associated with
6 each of the groups (Ga, Gb, Gc), each of said elementary queues (5a, 5b, 5c) storing a
7 predetermined number of tasks to be processed in a given order of priority, so that each of the

8 tasks of each of said elementary queues (5a, 5b, 5c) is associated with one of the processors of
9 this elementary queue (20a-21a, 20b-22b, 20c).

1 43. Architecture according to claim 42, further comprising means (6) for determining
2 the load of said elementary queues (5a, 5x, 5y, 5p) and for assigning a new task created in said
3 system to the elementary queue with the lightest load (5y).

*Sub
B1 1
2
3
A2 4
5*
44. Architecture according to claim 42, further comprising, when one (5q) of said
elementary queues (5a, 5x, 5y, 5p) associated with one of said processors (2q) is empty, means
(7) for locating a non-empty, remote elementary queue (5y), and an executable task in said non
empty elementary queue (5y), and assigning said executable task to said one of said processor
(2q) for processing said executable task.

45. Architecture according to claim 42, further comprising means (8) for detecting an
imbalance between elementary queues (5a, 5x, 5y, 5p), and for determining when an imbalance
is detected the elementary queue with the heaviest load (5x) and the elementary queue with the
lightest load (5y), and means for moving tasks from the elementary queue with the heaviest load
(5x) to the elementary queue with the lightest load (5y).

1 46. Architecture according to claim 42, wherein the operating system of the
2 processing system is of the nonuniform memory access type (NUMA), and comprises modules
3 (M0, M1) linked to one another, each module comprising a given number of processors (200-
4 203, 210-213) and storage means, each of said modules (M0, M1) constituting one of said
5 groups, each module (M0, M1) being associated with one of said elementary queues.

1 47. Architecture according to claim 43, wherein the operating system of the
2 processing system is of the nonuniform memory access type (NUMA), and comprises modules
3 (M0, M1) linked to one another, each module comprising a given number of processors (200-

4 203, 210-213) and storage means, each of said modules (M_0, M_1) constituting one of said
5 groups, each module (M_0, M_1) being associated with one of said elementary queues.

1 48. Architecture according to claim 44, wherein the operating system of the
2 processing system is of the nonuniform memory access type (NUMA), and comprises modules
3 (M_0, M_1) linked to one another, each module comprising a given number of processors (200-
4 203, 210-213) and storage means, each of said modules (M_0, M_1) constituting one of said
5 groups, each module (M_0, M_1) being associated with one of said elementary queues.

1 49. Architecture according to claim 45, wherein the operating system of the
2 processing system is of the nonuniform memory access type (NUMA), and comprises modules
3 (M_0, M_1) linked to one another, each module comprising a given number of processors (200-
4 203, 210-213) and storage means, each of said modules (M_0, M_1) constituting one of said
5 groups, each module (M_0, M_1) being associated with one of said elementary queues.--